



Using the Unity game engine as a platform for advanced real time cinema image processing

Timothée de Goussencourt, Pascal Bertolino

► To cite this version:

Timothée de Goussencourt, Pascal Bertolino. Using the Unity game engine as a platform for advanced real time cinema image processing. ICIP 2015 - 22nd IEEE International Conference on Image Processing, Sep 2015, Québec, Canada. hal-01208180v2

HAL Id: hal-01208180

<https://hal.science/hal-01208180v2>

Submitted on 24 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

USING THE UNITY® GAME ENGINE AS A PLATFORM FOR ADVANCED REAL TIME CINEMA IMAGE PROCESSING

Timothée de Goussencourt

SolidAnim,
GIPSA-lab, Grenoble Alpes University

*Pascal Bertolino **

GIPSA-lab, Grenoble Alpes University

ABSTRACT

The global context of our work is the virtual production film industry. We present an efficient framework to merge a low resolution depth map sensor with a high resolution film camera. The depth sensor used is a Kinect 2. Our method is especially designed for film production requiring live (pre) visualization. To achieve real-time performance we are not using a specific customized solution software but the very popular game engine *Unity*®. Our process is directly implemented into this game engine to give the user all the realtime facilities of a traditional game engine.

Index Terms— Depth map, Real-time, Compositing, Virtual Production, Previz on-set, *Unity*

1. INTRODUCTION

This article presents an efficient fusion method of a low resolution depth sensor with a high resolution film camera. Adding depth information to color is a huge benefit for film industry, especially for virtual production where real contents need to be mixed with digital assets (compositing). Different techniques that achieve this goal are presented in [1].

Some techniques to mix high definition color frames with depth sensors have already been developed. For instance [2] uses a trifocal rig to generate high precision depth maps: The authors combine a stereo system with a monocular depth sensor, similar to [3, 4, 5, 6]. Then, a complex global optimization workflow is needed to merge data. [7] uses a graph-cut optimization whereas [8] formulates a convex optimization problem to make depth image upsampling. All of these methods use custom rigs. In the scope of the *Scene* European project, the *Arri* company developed a prototype coupling a professional video sensor with a ToF depth sensor [9]. Conversely, the method we propose can deal with any film camera.

Our method calibrates sensors in a classic way and remaps the depth data into the main film camera referential. A depth comparison between real and virtual objects of the scene is performed to compose the final view. Existing custom implementations using depth cameras for mixed reality applications are already described in [10, 11]. Our main contribution is to demonstrate the use of a very popular game engine, *Unity*® (<http://unity3d.com/public-relations>) as a live previewing tool. With the later, scene edition is available in real-time and the user can benefit of all the advanced features available in a game engine to really focus on creativity for mixing virtual with real. To our best knowledge, our work is the first scientific publication about the use of a game engine in the field of real time special effects in cinema. This paper is outlined as follows: in 2 we discuss about important choices we made

for our system and compare them with existing works. Section 3 describes the main points of our method and how they are technically implemented into the game engine. The paper will be concluded with an experiment of our method in section 4.

2. BACKGROUND

A prerequisite of our processing is to get the depth information of the real scene. In the last years, several techniques have been developed to achieve this goal. Multicamera system around the scene can reconstruct a 3D volume of it. This approach is really effective but requires an important setup. Other approaches based on stereo vision could recover the depth of the scene but are limited by the amount of texture details available on the scene. To handle this limitation, some active stereo techniques have been developed based on structured light. The first generation of Kinect is based on this technique [12]. The use of an infrared pattern also has the advantage not to disturb the visible spectral domain of the scene. [13] builds a custom system mixing stereo vision with an infrared pattern projected to the scene. Another technique using infrared is the time of flight (ToF) technology in which the time that it takes for an infrared ray to hit an object of the scene and go back to the sensor is measured, providing the corresponding depth. This Kinect version is used in this work, as a good compromise of quality and price. Moreover Kinect 2 includes other interesting features like human pose recognition [14]. This sensor delivers a 512×424 pixels resolution, 16 bits per pixel real-time depth map. Contrary to existing works such as the one presented in [15], the color sensor available on the Kinect 2 is not used in our system. On the contrary, the Kinect is rigidly fixed to a high resolution color sensor with its lens (film camera) as shown in figure 2. Our current system is designed for indoor small scale scenes, in a controlled environment which correspond to many virtual production needs. The novelty of our fusion method lies in the use of the popular *Unity*® video game engine in which it is integrated. This platform was chosen because of its great versatility and huge developer community. The graphics engine incorporated into *Unity* is built on top of most modern graphics API, achieving real-time rendering performance, which was an important prerequisite for our project. In our case, the *Unity*® software is not used to build a game but to perform and display real-time image processing.

3. METHOD

Our method is divided into two main blocks. First a classical offline process retrieves the intrinsic parameters of each camera. Then an extrinsic calibration is done to compute the relative position of the depth sensor to the film camera. Second an online process is performed to back-project in real-time the 3D information from the

* The authors thank the French FUI for supporting the PREVIZ project.

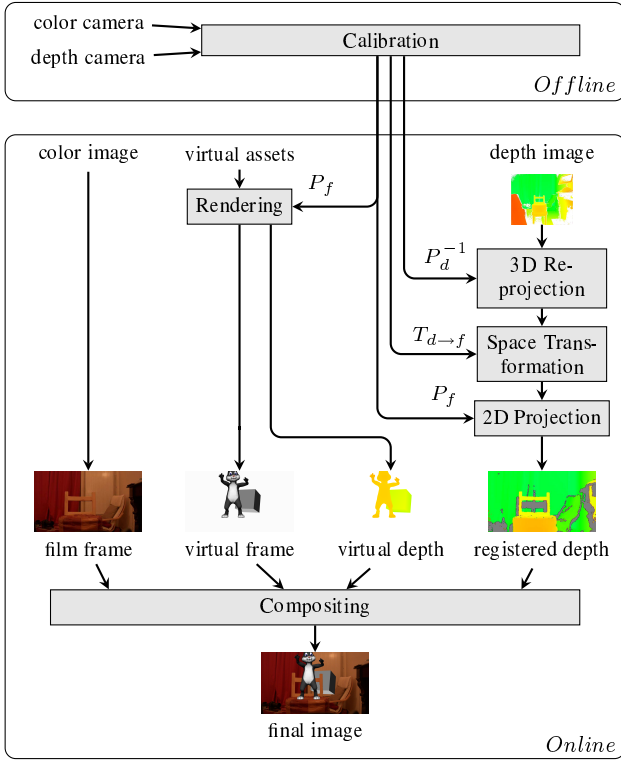


Fig. 1. Overall system workflow. Offline process is executed once at the beginning. Online process is executed in real-time for each frame.

depth sensor into the main camera screen space for each frame. Finally a compositing step mixes virtual and real assets. These two blocks are represented in figure 1. Each step of these blocks is presented in the sequel.

3.1. Calibration

One of the most important aspect of this application is the internal and relative calibration of the color camera and the depth camera. We consider the two sensors as classical perspective cameras. The first step of our method is to retrieve intrinsics and distortion parameters of each camera sensor. We use radial and tangential distortion models described in [16, 17] and a specific distortion model for fish-eye lenses described in [18]. A well-known technique for calibrating perspective camera is based on chessboard pattern recognition [19, 20]. This calibration is independently done for each camera, in color space for the film camera and in infrared for the depth sensor.

This leads to projection matrices P_d and P_f for respectively depth and film camera. A vector of distortion parameters is also computed for each camera. Whereas distortion for film camera completely depends of the used lens, it appears that Kinect 2 is very few distorted.

The calibration algorithm also gives for each chessboard view the pose of the chessboard relative to the camera system coordinates. Combining the relative pose of each camera to the chessboard makes it possible to compute for each view an estimation of the euclidean transformation $T_{d \rightarrow f}$ between the color camera and the depth camera. We take several captures of the scene with a chessboard at different depths, simultaneously with the depth and the color camera.



Fig. 2. Our system using different film cameras coupled with the Kinect 2. Left: Blackmagic Studio HD with 14mm lens. Right: Panasonic AG-AF100A with a 18mm lens.

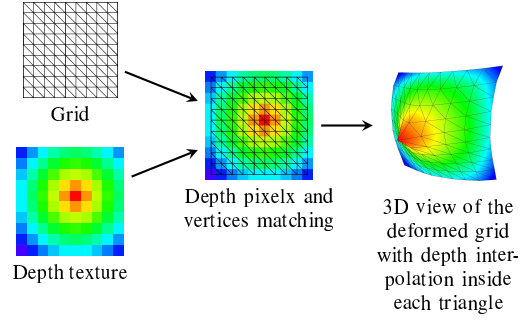


Fig. 3. 3D mesh reprojection method

Each couple of images is processed and the measurement of the current pose is added to a stack. Finally an overall adjustment is performed using a robust Levenberg-Marquardt iterative algorithm to obtain an optimal extrinsic calibration [21].

It is possible to retrieve intrinsics and extrinsic parameters in the same time but it is sometimes more appropriate to use two different sets of chessboards views. Indeed the covered field of view may differ between sensors. It is better to maximize the pattern size in the camera view to compute intrinsics and distortion parameters, whereas we need to see the same chessboard entirely from each sensor point of view for extrinsic calibration.

This calibration is computed only once at the beginning, all calibration parameters are saved and loaded by the online process.

3.2. Depth map reprojection into the film camera

Each step of the online process is implemented in *Unity*. Everything in this game engine relies on the concept of *gameobject*. In our case the film and depth cameras are represented by two camera gameobjects. The depth camera gameobject is parented to the film camera gameobject. The extrinsic calibration file computed during the offline process is loaded to correctly pose the depth camera object into the film camera object coordinates space.

Then a mesh gameobject is parented to the depth camera object. This mesh is a flat grid with the dimension of the depth map with $w \times h$ triangulated vertices. A $2D$ vector (u, v) is generated for each vertex to correctly associate its corresponding depth texture pixel. The GPU computation of the $3D$ scene is realized with a shader. For each vertex passed to the vertex shader, we look up into the depth texture according to the (u, v) coordinates to retrieve the associated depth data, $z = tex(u, v)$. Using the depth camera inverse projective matrix P_d^{-1} it is possible to compute the current $3D$ vertex position X_{3D} as detailed in equ. 1. Then a geometry shader discards the triangle primitives where no depth information

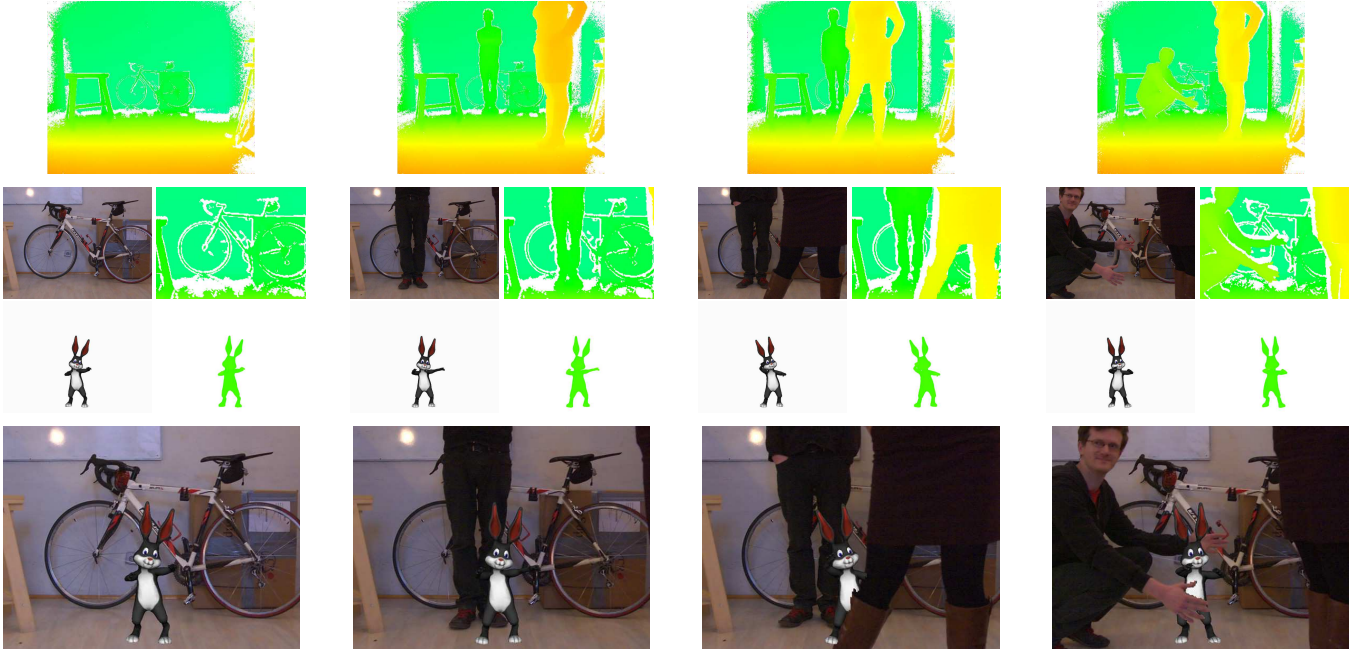


Fig. 4. Examples of compositing. First row: depth map. Second row: film frame and registered depth map. Third row: virtual character frame and depth map. Fourth row: final compositing. Occlusions are visible on third and fourth columns.

is present, to only display valid data. Finally a fragment shader is used to encode the interpolated depth value of the scene for each pixel. This way a 3D representation of the scene is computed in real-time using GPU capabilities. Figure 3 gives an illustration of the method.

$$X_{3D} = P_d^{-1} \cdot \begin{pmatrix} u \cdot tex(u, v) \\ v \cdot tex(u, v) \\ tex(u, v) \end{pmatrix} \quad (1)$$

Considering a grid mesh more than a list of 3D points has the advantage to observe a dense surface. This 3D mesh is now observed according to the film camera point of view using the projection matrix of the color camera P_f . Re-projection of the low resolution depth map becomes a dense interpolation fitting the resolution of the film camera. This observation is rendered to a texture called registered depth map.

3.3. Compositing

Then it is possible to compare the real scene depth map with the virtual scene depth map to generate a binary mask coding whether a pixel of the final rendering comes from the virtual asset frame or from the film camera frame. This way, original color image could be associated with its registered and distorted depth image, making compositing possible, whatever occlusion occurs between the virtual and real worlds.

4. EXPERIMENTS

Our system was tested with two different film cameras used traditionally for broadcast applications (see Figure 2). A live video stream is transferred from the camera to the computer with an SDI to USB3 acquisition card. The computer is a laptop with an Intel i7 2.3GHz processor, 16Go RAM and a Nvidia GeForce GTX 670M.

We use the special *Unity* plugin *AVPro Live Camera* to pipe the HD color frames into *Unity*. Moreover *Microsoft* provides plugins to access Kinect 2 functionalities into the game engine. For each camera, a frame buffer is set up to compensate the small delay between depth and color streams. Figure 4 shows our result obtained with a 18mm lens mounted on a Panasonic AG-AF100A. In this configuration the pixel per degree ratio between the film and the depth camera gives us a scale factor of about 5 on each dimension. This means a pixel on the original depth map influences a 5×5 pixels patch on the film image. The average time consumed by the overall on-line process for one frame is about 3ms saving a lot of time for any other processing. Some live video samples captured on the laptop screen are available at <https://sites.google.com/site/pascalbertolino/previz>.

5. CONCLUSION

We presented a reliable method to align a Kinect 2 depth sensor with a professional film camera. This method has been implemented into the popular game engine *Unity*® and achieves realtime performances. This system is designed for indoor small scale scenes and is generic to any film camera. The framework we described here is used in the scope of the PREVIZ project [22] dedicated to virtual production, especially to previz on-set. In order to enlarge the compositing scenario range, we coupled our system to the *SolidAnim* camera tracking solution (SolidTrack). Thus it was straightforward to remove real background and replace it with a virtual one without using green or blue walls. It was easy as well to display a virtual character or object in the scene while respecting any occlusion between real and virtual worlds. In the future, using *Unity* will allow to simply make powerful and fast process with the GPU, like improving the quality and regularity of the depth maps or managing interactions between the virtual world and a human body.

6. REFERENCES

- [1] G. Thomas, “Mixed reality techniques for tv and their application for on-set and pre-visualization in film production,” in *International Workshop on Mixed Reality Technology for Film-making*, 2006, pp. 31–36.
- [2] G. Boisson, P. Kerbirou, V. Drazic, O. Bureller, N. Sabater, and A. Schubert, “Fusion of kinect depth data with trifocal disparity estimation for near real-time high quality depth maps generation,” in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2014, pp. 90110J–90110J.
- [3] J. Zhu, L. Wang, R. Yang, and J. Davis, “Fusion of time-of-flight depth and stereo for high accuracy depth maps,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.
- [4] U. Hahne and M. Alexa, “Combining time-of-flight depth and stereo images without accurate extrinsic calibration,” *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3, pp. 325–333, 2008.
- [5] Y. Song, C.A. Glasbey, G.W. van der Heijden, G. Polder, and J.A. Dieleman, “Combining stereo and time-of-flight images with application to automatic plant phenotyping,” in *Image Analysis*, pp. 467–478. Springer, 2011.
- [6] V. Gandhi, J. Cech, and R. Horaud, “High-resolution depth maps based on tof-stereo fusion,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4742–4749.
- [7] S. Patra, B. Bhowmick, S. Banerjee, and P. Kalra, “High resolution point cloud generation from kinect and hd cameras using graph cut,” in *VISAPP (2)*, 2012, pp. 311–316.
- [8] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof, “Image guided depth upsampling using anisotropic total generalized variation,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 993–1000.
- [9] Thomas Hach and Johannes Steurer, “A novel rgb-z camera for high-quality motion picture applications,” in *Proceedings of the 10th European Conference on Visual Media Production*, New York, NY, USA, 2013, CVMP ’13, pp. 4:1–4:10, ACM.
- [10] B. Bartczak, I. Schiller, C. Beder, and R. Koch, “Integration of a time-of-flight camera into a mixed reality system for handling dynamic scenes, moving viewpoints and occlusions in real-time,” in *Proceedings of the 3DPVT Workshop, Atlanta, GA, USA (June 2008)*, 2008.
- [11] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al., “Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera,” in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.
- [12] A. Shpunt and Z. Zalevsky, “Three-dimensional sensing using speckle patterns,” Mar. 5 2013, US Patent 8,390,821.
- [13] M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, et al., “Real-time non-rigid reconstruction using an rgb-d camera,” *ACM Transactions on Graphics, TOG*, vol. 4, 2014.
- [14] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.
- [15] M. Camplani, T. Mantecon, and L. Salgado, “Depth-color fusion strategy for 3-d scene modeling with kinect,” *Cybernetics, IEEE Transactions on*, vol. 43, no. 6, pp. 1560–1571, 2013.
- [16] D.C. Brown, “Close-range camera calibration,” *Photogramm. Eng.*, vol. 37, pp. 855–866, 1971.
- [17] J.G. Fryer and D.C. Brown, “Lens distortion for close-range photogrammetry,” *Photogrammetric engineering and remote sensing*, vol. 52, no. 1, pp. 51–58, 1986.
- [18] F. Devernay and O. Faugeras, “Straight lines have to be straight,” *Machine vision and applications*, vol. 13, no. 1, pp. 14–24, 2001.
- [19] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. IEEE, 1999, vol. 1, pp. 666–673.
- [20] Z. Zhang, “A flexible new technique for camera calibration,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.
- [21] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*, ” O’Reilly Media, Inc.”, 2008.
- [22] G. Briand, F. Bidgolirad, J.F. Zlapka, J.M. Lavalou, M. Lanouiller, M. Christie, J. Lvoff, P. Bertolino, and E. Guilou, “On-set previsualization for vfx film production,” in *International Broadcasting Convention (IBC)*, Amsterdam, Netherlands, 2014.